Control Strategies for Resolution.

1. **Breadth-First Strategy**: All of the first-level resolvents are computed first, then the second-level resolvents, and so on. An i-th level resolvent is one whose deepest parent is an (i-1)-th level resolvent. The strategy is complete.
2. **Set-of-Support Strategy**: At least one parent of each resolvent is selected from among the clauses resulting from the negation of the goal or from their descendants (set of support). The strategy is complete.
3. **The Unit-Preference Strategy**: The same as set-of-support but we try to select a single-literal clause to be a parent in a resolution. The strategy is complete.
4. **The Linear-Input Form Strategy**: Each resolvent has at least one parent belonging to the base set. The strategy is not complete.
5. **The Ancestry-Filtered Form Strategy**: Each resolvent has a parent that is either in the base set or that is an ancestor of the other parent. The strategy is complete.

Problem 1.

Knowledge Base

Whoever can read is literate: $(\forall x)[R(x) \rightarrow L(x)]$

Dolphins are not literate: $(\forall x)[D(x) \rightarrow \sim L(x)]$

Some dolphins are intelligent: $(\exists x)[D(x) \wedge I(x)]$

Prove that: Some who are intelligent cannot read: $(\exists x)[I(x) \wedge \sim R(x)]$???

Breath First Strategy

~R(x) v L(x)   ~D(x)v ~L(x)   D(A)   I(A)   ~I(x) v R(x)

~R(x)v ~D(x)   ~L(A)   R(A)   L(x)v ~I(x)

~I(x)v ~D(x)   ~R(A)   L(x)   ~D(A)

~D(x)   NIL

Set-of-Support Strategy (clauses marked by red form set of support)

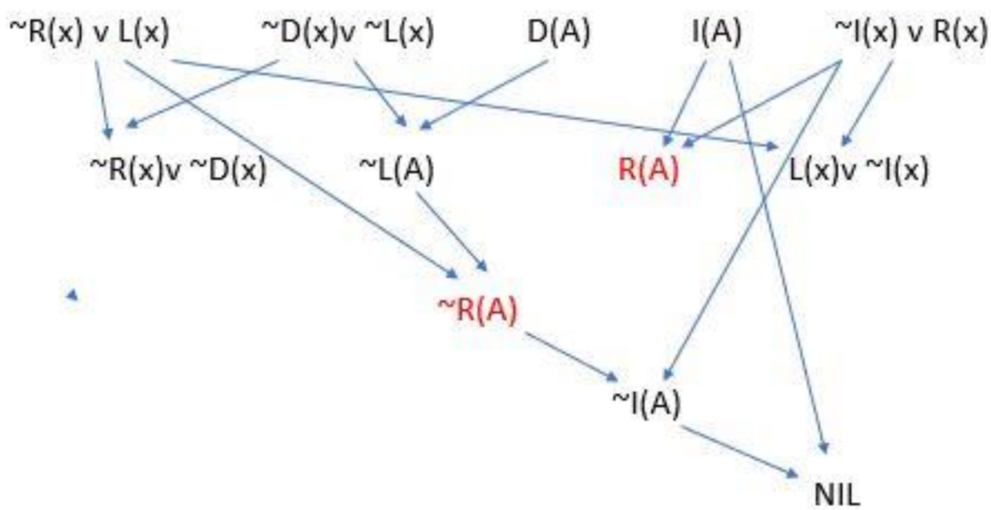
~R(x) v L(x)   ~D(x)v ~L(x)   D(A)   I(A)   ~I(x) v R(x)

~R(x)v ~D(x)   ~L(A)   R(A)   L(x)v ~I(x)

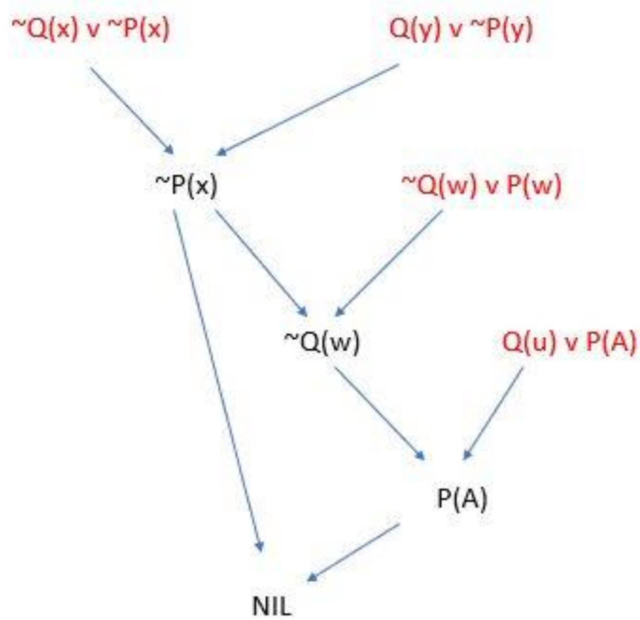
~I(x)v ~D(x)   ~R(A)   L(A)   ~D(A)   ~I(A)

~D(x)   NIL

The Ancestry-Filtered Form Strategy (?)



Example of Ancestry-Filtered Form Strategy (using Merge) **PROLOG**



We use Merge
(preserves completeness)

Resolution which is also
replacing Q v Q by Q

Problem 2

Consider the following facts:

(1) Every child loves anyone who gives the child any present.

(2) Every child will be given some present by Santa if Santa can travel on Christmas eve.

(3) It is foggy on Christmas eve.

(4) Anytime it is foggy, anyone can travel if he has some source of light.

(5) Any reindeer with a red nose is a source of light.

Prove that: If Santa has some reindeer with a red nose, then every child loves Santa.

Alphabet: child(x) – x is a child, loves(x,y) – x loves y, gives (x,y) – x gives y a present, travel(x) – x can travel on Christmas, Foggy – foggy on Christmas Eve, has(x, y)- x has y,
SL-source of light, RRN- reindeer with a red nose.

Knowledge Base:

[gives(x,y) ^ child(y)] -> loves(y,x)
[travel(Santa) ^ child(y)] -> gives(Santa, y)
Foggy
Foggy ^ has(x,SL) -> travel(x)
has(x, RRN) -> has(x, SL)
Prove: has(Santa, RRN) ^ child(y) -> loves(y, Santa) ????

Clauses

~gives(x,y) v ~child(y) v loves(y,x)
~travel(Santa) v ~child(y) v gives(Santa, y)
Foggy
~Foggy v ~has(x,SL) v travel(x)
~has(x, RRN) v has(x, SL)
Prove: ~has(Santa, RRN) v ~child(y) v loves(y, Santa)

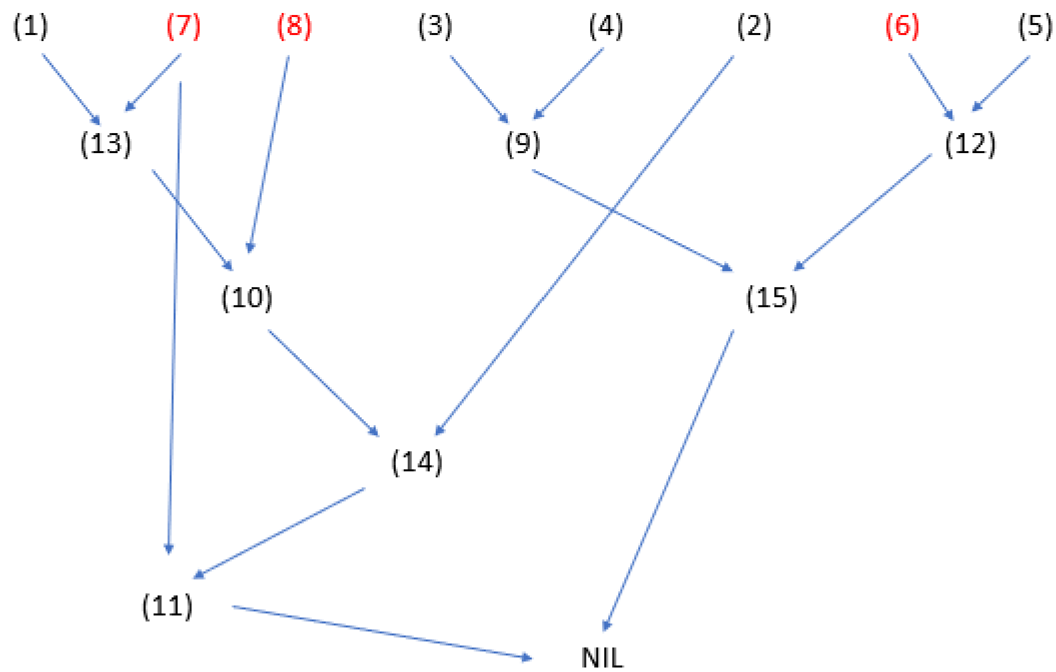
Clauses

1) ~gives(x,y) v ~child(y) v loves(y,x)
2) ~travel(Santa) v ~child(y3) v gives(Santa, y3)
3) Foggy
4) ~Foggy v ~has(x2,SL) v travel(x2)
5) ~has(x1, RRN) v has(x1, SL)
6) has(Santa, RRN)
7) child(y1)
8) ~loves(y2, Santa)

Resolution

(9)=(3)+(4): ~has(x2,SL) v travel(x2)
(13)=(1)+(7): ~gives(x,y) v loves(y,x)
(10)=(13)+(8): ~gives(Santa,y)
(14)=(10)+(2): ~travel(Santa) v ~child(y3)
(11)=(14)+(7): ~travel(Santa)
(12)=(5)+(6): has(Santa, SL)
(15)=(9)+(12): travel(Santa)
(15)+(11): NIL

Resolution (what kind of resolution strategy?)



Problem. Convert to the set of clauses:

(∃z)[(∃x)Q(x, z) → (∃x)P(x)] → ¬{¬(∃x)P(x) → (∀x)(∃z)Q(z, x)}

(∀z)[(∃x)Q(x, z) ^ (∀x)~P(x)] v {~(∃y)P(y) ^ ~(∀w)(∃s)Q(s, w)}

[Q(x(z), z) ^ ~P(t)] v [~P(y) ^ ~Q(s, w(y))] .